Aprendiendo “Git” && “GitHub”

{

[03 Ramas - Branches]

< Ramas (branches) líneas del tiempo paralelas y simultáneas en un proyecto />

}

Contenido {

}

A.rtf

A {

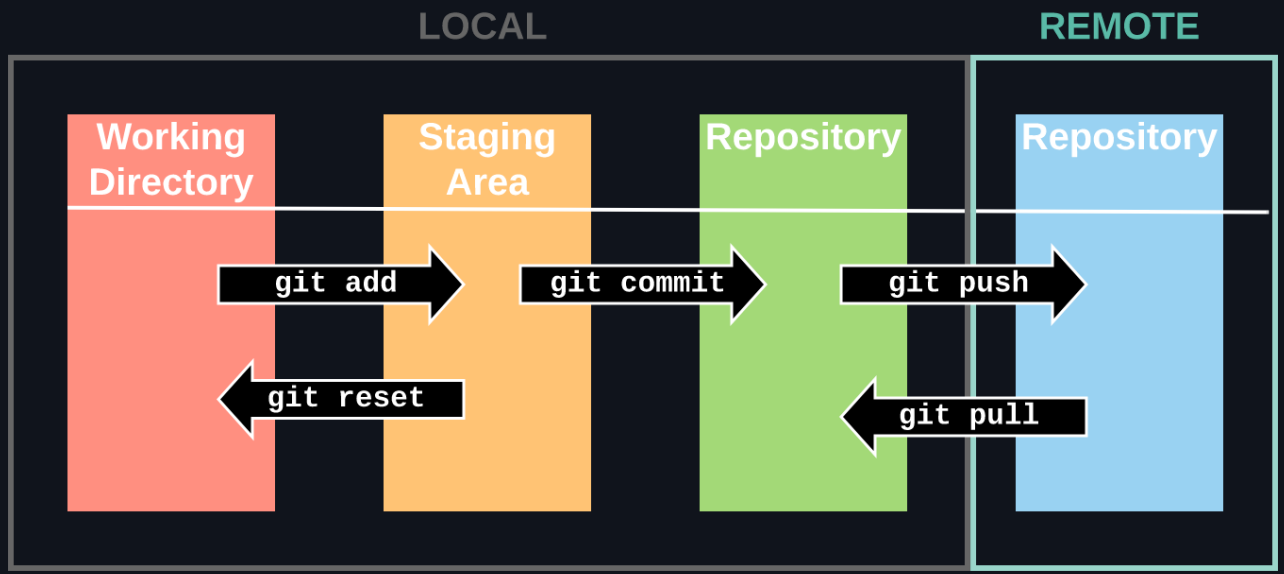
[Recapitulación]

< Comandos de navegación
en la línea del tiempo />

}

A.rtf

git-flow.exe



A. Recapitulación

[1] Inicializar un repositorio:

[git init]

Debe hacerse este comando mientras se está en la **carpeta principal** del proyecto;

OK

[2] Revisar el estatus del repositorio:

[git status]

NOTA: Debe intentarse leerse y entender el flujo de los archivos;

OK

[3] Preparar cambios de un archivo para antes de hacer commit (al staging):

[git add archivo.txt]

Esto para decir qué archivos queremos registrar en un punto de tiempo (commit);

OK

[4] Agregar los cambios de todos los archivos:

[git add .]

Lo mismo que el anterior, pero este prepara los cambios en todo el documento para hacer commit;

OK

[5] Registra un “PUNTO DE TIEMPO” en el repositorio los cambios agregados y preparados en la zona del *staging*:

[git commit -m “Se actualizó el algoritmo para arreglar errores”]

Deja un punto en la línea del tiempo en donde se deja registro de los cambios que agregamos con **“git add”**;

OK

[6] Registrar cambios con doble mensaje (título del commit y descripción del commit):

[`git commit -m` “Se actualizó el algoritmo para arreglar errores”]

Lo mismo que el anterior, pero permitiendo dejar un mensaje de título y uno de descripción...

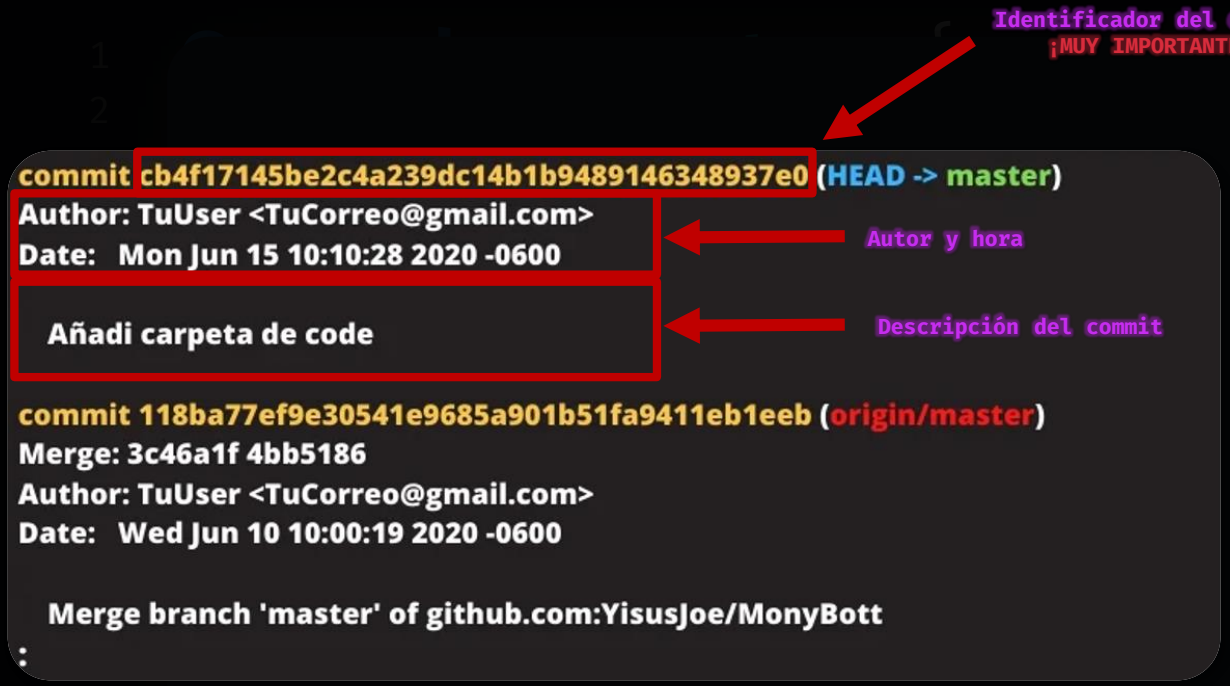
OK

[7] Ver línea de tiempo de cambios:

[git log]

Muestra la línea de historia de commits con la siguiente información…

OK



Entiéndase que cada “commit” es el grupo de cambios que enviamos al repositorio.

Se recomienda hacer commit cada que queramos quedar registro de avance o cambios, con alguna descripción significativa…

```
commit cb4f17145be2c4a239dc14b1b9489146348937e0 (HEAD -> master)
Author: TuUser <TuCorreo@gmail.com>
Date:   Mon Jun 15 10:10:28 2020 -0600

    Añadi carpeta de code

commit 118ba77ef9e30541e9685a901b51fa9411eb1eeb (origin/master)
Merge: 3c46a1f 4bb5186
Author: TuUser <TuCorreo@gmail.com>
Date:   Wed Jun 10 10:00:19 2020 -0600

    Merge branch 'master' of github.com:YisusJoe/MonyBott
:
```

Algunos commits nos mostrarán datos de su posición.

HEAD significa en qué punto del tiempo estamos actualmente;

HEAD -> master significa que actualmente estamos situados en el tope de la rama “master”…

[9] Viajar en el tiempo a un commit pasado (suave):

[git reset #commitID --soft]

El git reset **(suave: soft)** regresa todo el proyecto a un commit especificado, pero el **staging seguirá conservando** cambios que hayamos agregado…

OK

El “**git reset soft**” es como **viajar a un punto del tiempo** en el pasado, pero **conservando la máquina del tiempo** que nos permita regresar.

OK

[10] Viajar en el tiempo a un commit pasado **(DURO!)**:

[git reset #commitID --hard]

Precaución con este comando, pues **regresará todo el proyecto al commit que especifiquemos** y **borrará toda la historia** que haya quedado por enfrente de este commit…

OK

El “**git reset hard**” es como **regresar al pasado** pero **sin la posibilidad de regresar,** quedándose sin máquina del tiempo en el punto al que se llegó…

OK

[11] Viajar al tope de una rama:

[git checkout <larama>]

Indispensable para navegar entre ramas…

RECOMENDABLE USAR ESTE COMANDO PRIMERO HACIENDO COMMIT A LOS CAMBIOS…

OK

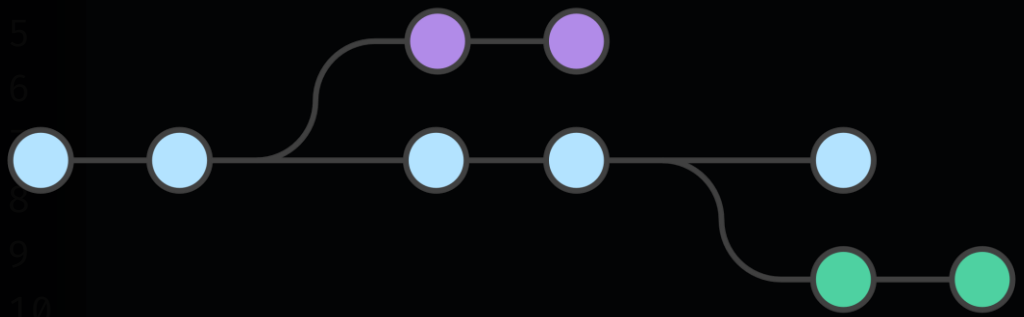
B.git

```
B {

  [Ramas (Branches)]

    < ¿QUÉ CA***O son las
    RAMAS en Git? />

}
```



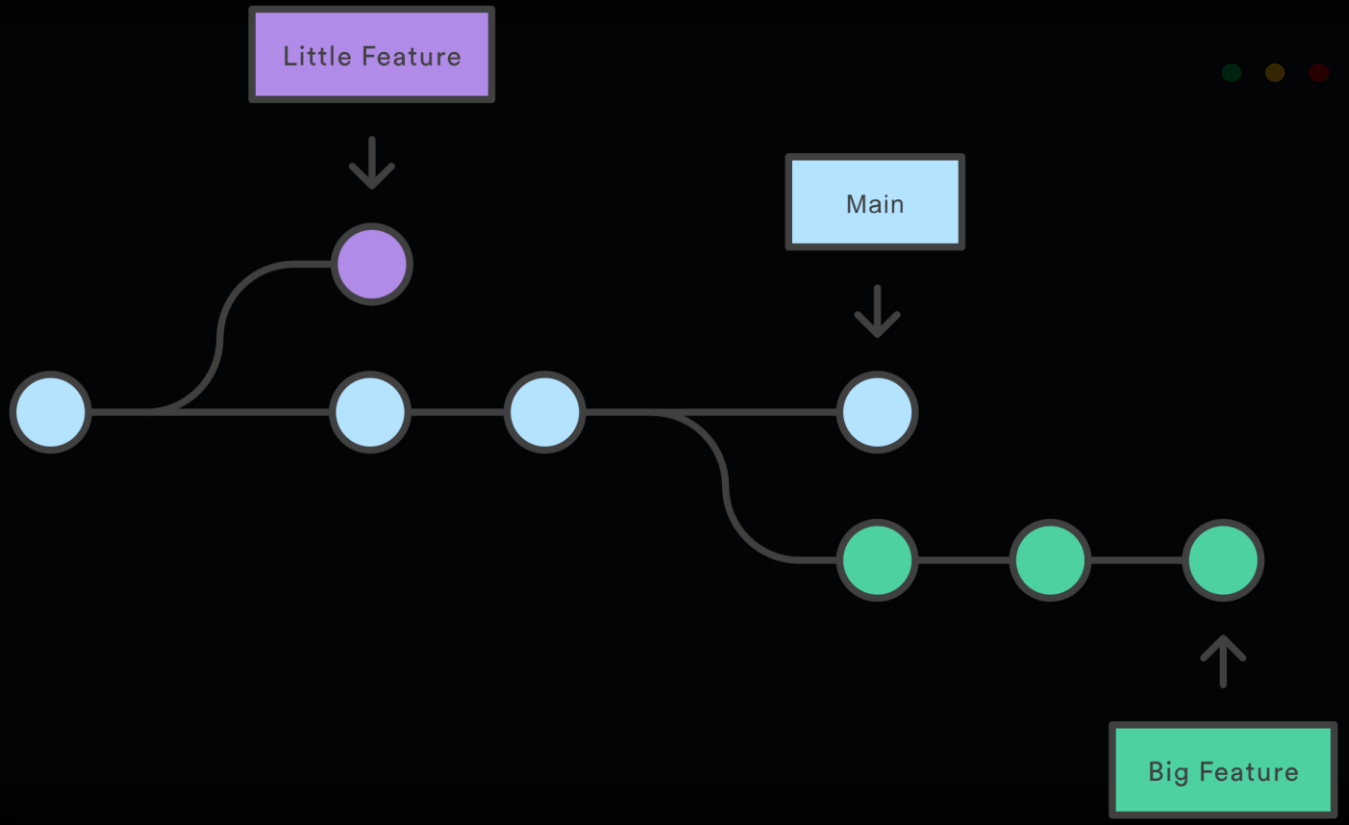
Si con git podemos controlar una “Línea del tiempo” que registra en cada punto (commit) el cambio hecho en el proyecto…

Las ramas son como tener líneas del tiempo paralelas a la línea original, en la que cada una tiene sus propias variaciones…

OK

B.git

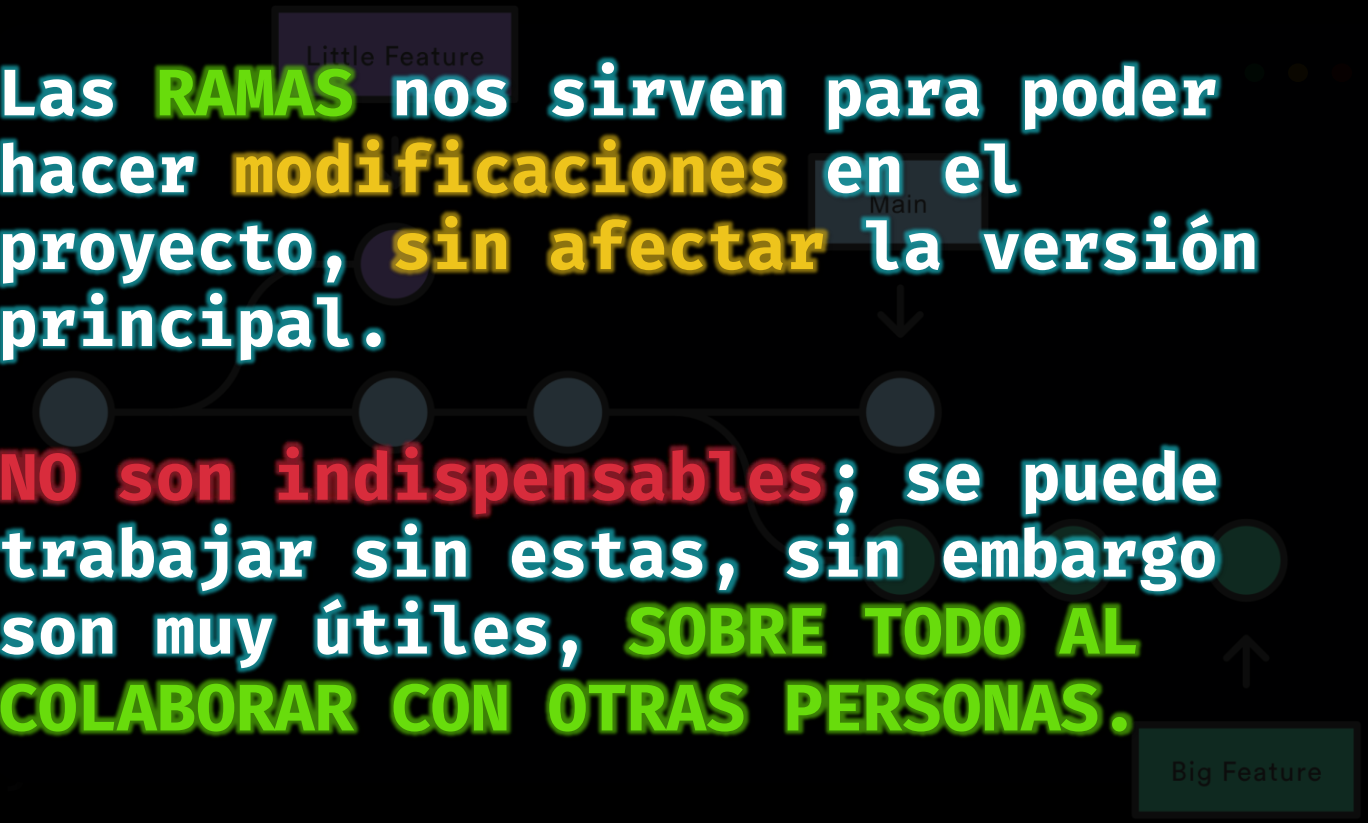
branches.md



B. Ramas (Branches)

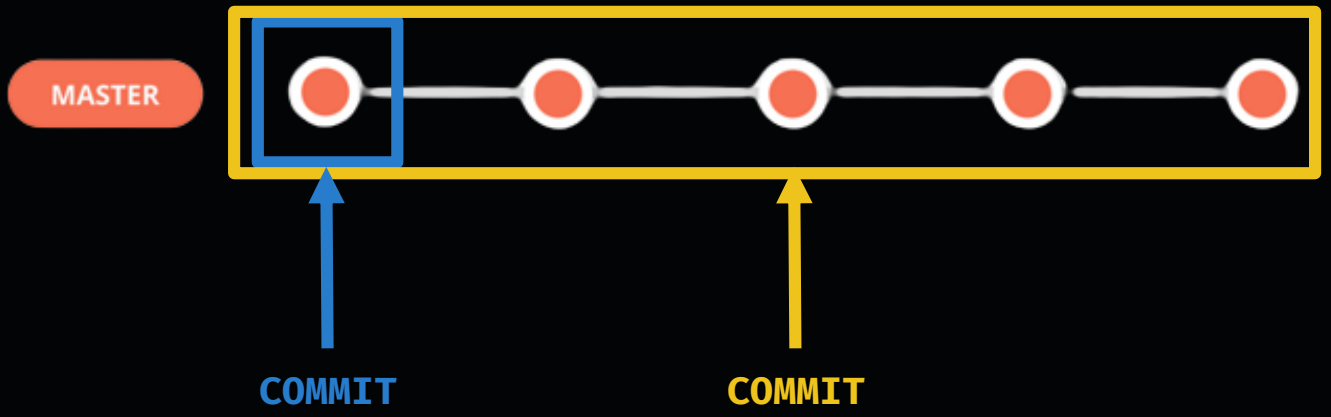
Las **RAMAS** nos sirven para poder hacer **modificaciones** en el proyecto, **sin afectar** la versión principal.

NO son indispensables; se puede trabajar sin estas, sin embargo son muy útiles, **SOBRE TODO AL COLABORAR CON OTRAS PERSONAS.**

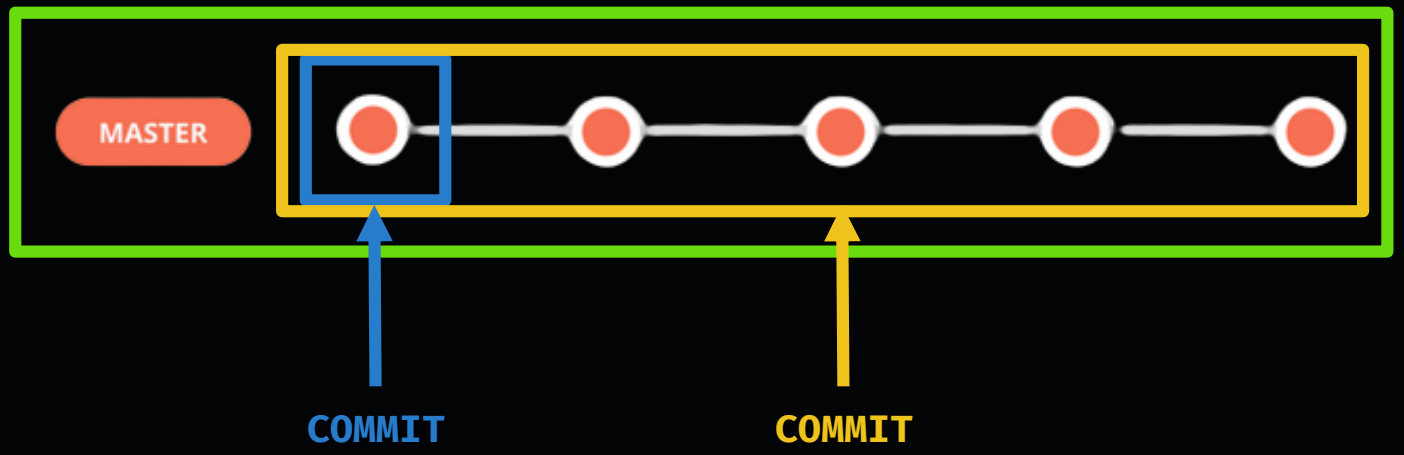


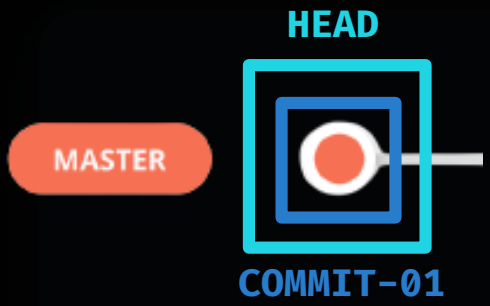
Imaginando que tenemos nuestra línea del tiempo de commits “MASTER” es la rama principal por defecto en Git;

Tenemos que:

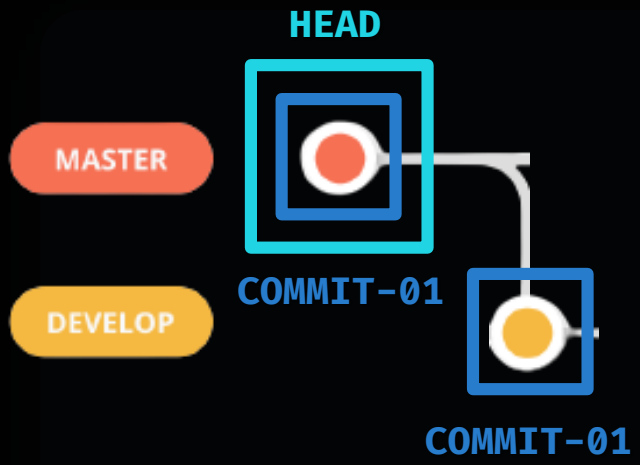


RAMA





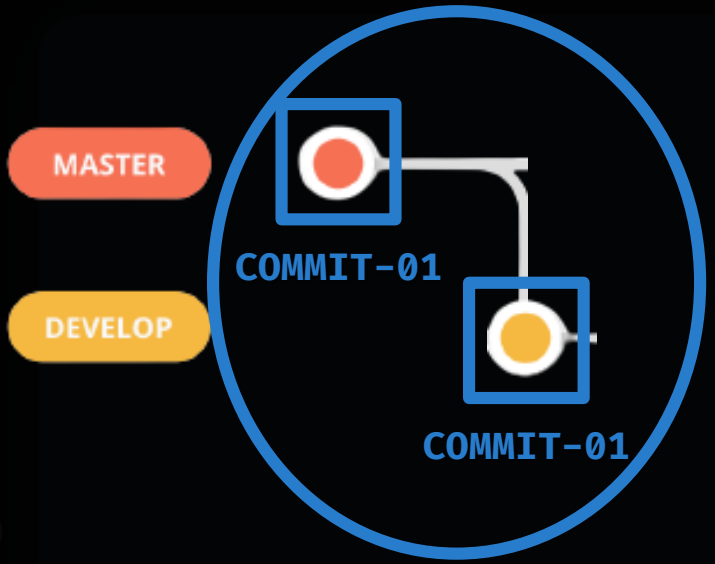
Al crear una rama, lo que hacemos es crear una COPIA del último commit de la rama en la que estamos (master), y esa copia de commit ahora está en la nueva rama que creamos…



Estando en “master”; el comando:

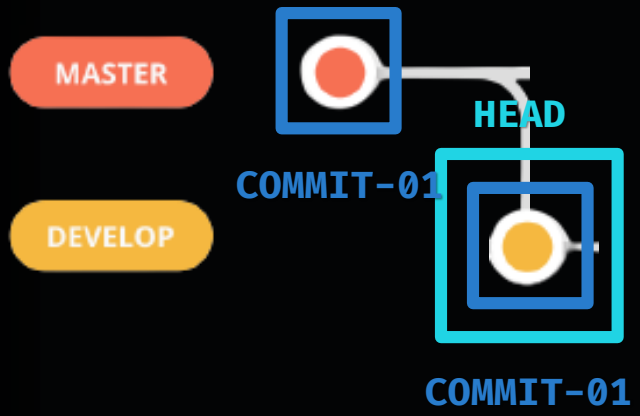
[git branch develop]

Crea una rama que hemos llamado “develop”, en la que se coloca el commit desde el que hicimos el comando (HEAD), que es el *commit-01* de la rama master…



Ambos commits son el mismo.

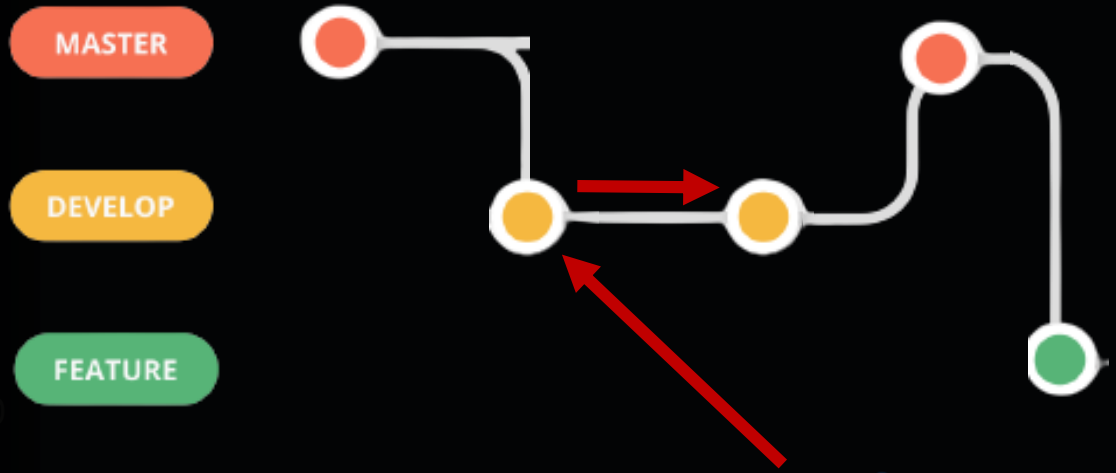
NOTA: REVISAR DESDE GIT BASH LO QUE SE MUESTRA;



Con el comando:

[git checkout develop]

Movemos el "HEAD" (El estado actual del proyecto), al commit más nuevo de la rama que especificamos (develop);

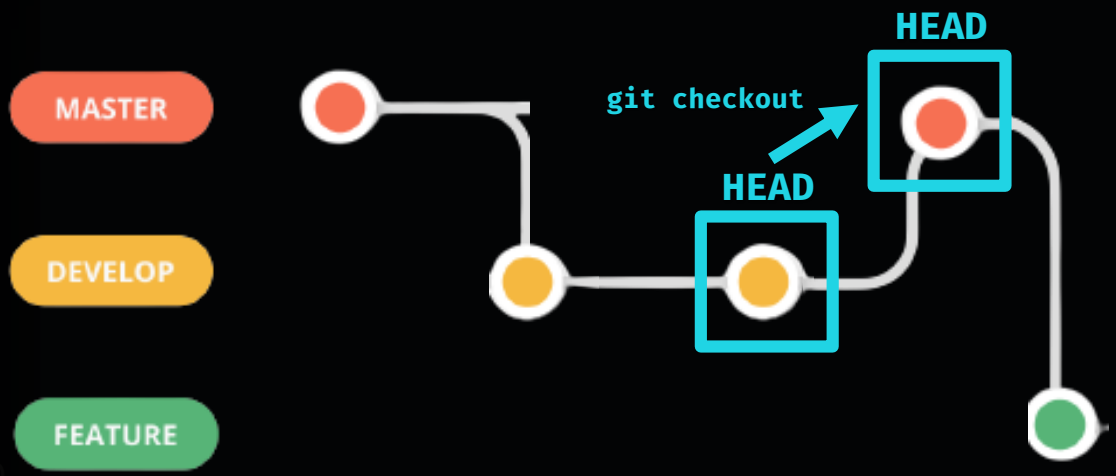


Al hacer commit desde la rama “develop”, el siguiente commit continúa desde la misma rama, y “master” continúa existiendo como estaba.

Comando IMPORTANTE:

[git merge develop]

Haciendo el comando desde “master”, mezclamos los cambios de “develop” en master.



Mover el “HEAD” con git checkout lleva todo el documento al punto en que se envíe, y permite regresar mientras no se modifique la línea de tiempo con más *“commits”*.

Explicación choncha dada en persona…